236371 Project in Distributed Programming – Serverless Classifier on OpenWhisk

High Level Design

Assumptions and design constraints

In this project we will implement a serverless image classifier using OpenWhisk. The classifier will be accessible through a custom website where the user can upload an image that they want to be classified. The website will encode the image and send it to the serverless classifier. Once the image is classified, the result will be returned to the user's page and displayed there.

We limit ourselves to JPG images and HTTP REST APIs to enable us to successfully build this project in the given amount of time. We also currently plan on using the NasNetMobile model that was trained on ImageNet and classifies an image from 1,000 different classes. This also means we plan on using Python 3.7 and TensorFlow. We also assume the user has access to an up-to-date browser like Chrome or Firefox and has a connection to the internet. This will be used to browse the website and access the serverless function which will both be hosted on the IBM cloud.

Many of these limitations and constraints could be relaxed later in the project's development cycle or after the first version is finished, as needed.

Dependencies

- TensorFlow and Keras
- The NasNetMobile model
- Python 3.7
- IBM Cloud Functions (uses OpenWhisk)
- IBM S3 hosting
- IBM DNS service
- Docker and the IBM Cloud Function Python Docker image
- HTML5, Javascript and Bootstrap 5.0

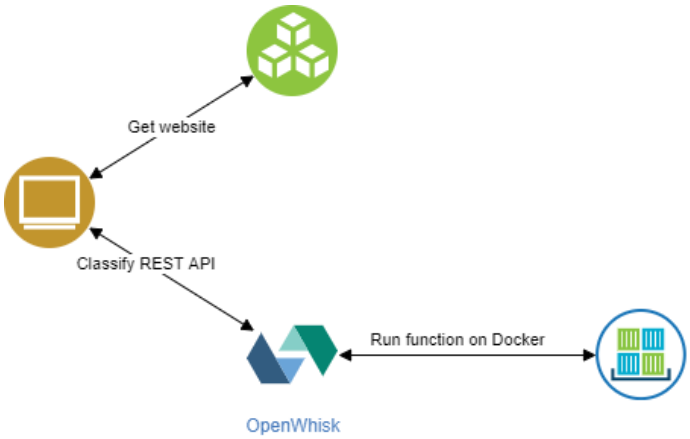
Open Issues

- Create a docker image with the classification function
- Implement the serverless classification function and make it accessible with a REST API
- Develop a website that uses the function
- Expose information and statistics on the serverless function

TODO list (w/ timetable)

- Create the docker image – 4.6
- Implement the serverless function – 11.6
- Develop the website – 18.6
- Expose function information – 25.6
- Extra features – 14.7

Logical Architecture



The website will be hosted in the IBM cloud, and will provide a basic UI for accessing the classification function. By uploading the image and requesting a classification, the website will encode the image as a JSON object and send it via a REST API to the IBM Cloud Functions gateway (which is based on OpenWhisk). The OpenWhisk instance will run the function with the received image on an existing docker image or create one if needed. It will return the result as a JSON object to the website and show the information to the user.

Additionally, the website will have an admin panel that shows performance information and statistics on the IBM Cloud Functions invocations, runtime etc. These will be requested from the IBM Metrics RST API and presented on the website.

Design

There aren't many classes in this project, so we will describe the main modules that we will use:

- IBM Cloud Functions – This service is based on OpenWhisk. It allows the user to create a function using some existing runtime environments or a docker image that exposes a specific function. We use the docker option because the supported runtimes don't have the libraries that we need in order to use a pre-trained image classification model. This service also exports a REST API for invoking a cloud function.
- Docker – We base our docker image on the IBM Cloud Function Python 3.7 action image published here. We install the needed libraries such as TensorFlow and Numpy on the image, so that when a new container is created it will have all the libraries that are needed. We also add the pre-trained model to the docker image, so it won't have to be downloaded every time. Additionally, we include a python script to do the actual classification.
- Python Script – The python script uses the libraries and the pre-trained model in the docker container to classify a received image and return the result. It handles all of the errors and reports accordingly. The script must export a main() function that receives a dict object and returns a dict object. OpenWhisk invokes this function on every invocation of the cloud function to run the requested action. We use the dict objects to send the image and receive its classification.
- Website – The website is the gateway for the user to the classification service. On the website, the user can upload an image and receive its classification. Once an image is uploaded, the website encodes the image locally and sends it in a JSON object to the IBM Cloud Function's REST API. It then waits for a response from the cloud function service, and once it receives the response it presents it to the user. This is done using AJAX requests that are included in the

website's javascript. Additionally, a separate page shows information on the performance and statistics of the IBM Cloud Functions service using the IBM Monitor REST API.

Setup and Installation Process

The installation process includes two steps: setting up the cloud function and hosting the website.

For the first step we must create a docker image that is based on the IBM Cloud Functions Python 3.7 Docker image. We build a new image that contains all the needed libraries (such as TensorFlow) and the pre-trained model. We also add a python script that exposes the function that will be run by OpenWhisk – this will be our classification function.

After the image is created, we can create a new IBM Cloud Function using the IBM OpenWhisk command line tool. We specify the details of the function, such as the docker image to be used and the name of the function. When this is done, we can start invoking the function and test it out.

The second step is deploying the website. Once we have a working website locally, we can host it on the IBM Cloud S3 service. We need to upload all the files and place them in their proper locations. Then we can connect it to a useful name using a DNS service and domain, to help users find it easily. Finally, we can browse the website and check out all the features.

This installation process is probably similar in other cloud environments as well, such as AWS or Microsoft Azure. They also have cloud functions and S3 object hosting that work similarly to the services we used.

Main Use Cases

The main use case of this project is classifying a given image from the user. The user uploads an image and receives a classification of the image from up to 1,000 different classes that the model was trained on. Additionally, a user can observe the metrics and performance of the IBM Cloud Functions instance.

This is done to learn more about OpenWhisk and serverless functions and present a way to host a website with meaningful use without running any virtual server. The website will be hosted statically in an S3 bucket, and the classification is implemented as a serverless function. Thus, the cost of providing the classification service scales only according to its use and does not have any additional overhead costs for installation or maintenance.

External Resources

- Learning Transferable Architectures for Scalable Image Recognition
- IBM Cloud Docs
- Bootstrap Docs